Information, Computing and Intelligent Systems, 2026, No. 8, 36 – 50
UDC 004.2:004.31 https://doi.org/10.20535/2786-8729.8.2026.356182

A METHOD FOR ACCELERATING PERCEPTRON COMPUTATIONS ON FPGA BASED ON ONLINE ARITHMETIC FOR CONVOLUTIONAL NEURAL NETWORKS

Illia Verbovskyi*
https://orcid.org/0009-0008-4782-4281

Valerii Zhabin
https://orcid.org/0000-0003-0377-3394

National Technical University of Ukraine
"Igor Sikorsky Kyiv Polytechnic Institute", Kyiv, Ukraine

*Corresponding author: illyaverb@gmail.com

Received: 10 Jan 2026 / Accepted: 15 March 2026 / Published: 28 May 2026

Approaches to accelerating result formation in a perceptron implemented on a field-programmable gate array through the application of online arithmetic in a redundant number system have been investigated. The classical parallel implementation of a neural computational path exhibits nonlinear latency growth as operand bit-width n and input count N increase. That growth reflects carry propagation in wide adder trees. The object of study is the complete hardware computational path of the perceptron from parallel multiply operations through the adder reduction tree to the threshold activation function. Its integration as a classifier block within a convolutional neural network is examined as well. The aim is to design and verify an architecture that enables overlapping of dependent computation stages while reducing the load on critical hardware resources. Compared to the parallel approach, a deterministic output-latency profile is still preserved. The methodological basis encompasses closed-form latency modelling with the online delay parameter p. Synthesis covers online multipliers with redundant coding and 3:2 reduction trees. Active-HDL functional and timing simulation and Quartus place-and-route target Altera Cyclone III EP3C5E144. For $N = 64$ and $n = 16$ at 200 MHz, the proposed online approach attains a 2.29 shorter result-formation interval than the in-house parallel baseline. First-level information load drops by 62.5%. Logic blocks are reduced from 64 to 24 at the cost of a modest flip-flop increase (9180 to 9720). The scientific novelty is a unified, experimentally verified architectural model that combines redundant online arithmetic with a realistic neural network inference context. It simultaneously reduces latency, lowers elements demand, and sustains deterministic real-time operation.

Keywords: field-programmable gate array, perceptron, online arithmetic, redundant number system, convolutional neural network, hardware acceleration, dependent computation chains, real time.

1. Introduction

Modern embedded intelligent data-processing systems are applied in classification, machine vision, and sensor analytics tasks. They require not only sufficient throughput but also predictable and deterministic decision-formation time under tight hardware resource constraints. Architectural design of such systems on field-programmable gate (FPGA) is governed by the balance among core latency, die area and power consumption. For perceptron implementation on FPGA, the key challenge is the efficient computation of chains of dependent operations. In the classical parallel approach, growth of operand bit-width n and input count N leads to increased latency in adder nodes due to mandatory carry propagation. Substantially more complex routing follows, and this trend is particularly critical for resource-constrained devices.

This problem formulation is directly linked to the practical requirements of hard real-time data-processing systems. Core latency determines control-loop quality, classifier robustness under changing conditions, and the maximum admissible input-stream rate. A delay exceeding one or two clock cycles from data arrival to output may violate hard timing guarantees in embedded systems. Online

© The Author(s) 2026. Published by Igor Sikorsky Kyiv Polytechnic Institute. This is an Open Access article distributed under the terms of the license CC BY 4.0 (https://creativecommons.org/licenses/by/4.0/), which permits reuse, distribution, and reproduction in any medium, provided the original work is properly cited.

arithmetic in a redundant number system potentially eliminates the need to wait between successive operations, since each subsequent operation can begin. After receiving only a few most significant digits of the previous result, creating conditions for natural overlapping of dependent computation stages [1].

This paper examines two alternative architectures – parallel and online – within a unified metric framework for the perceptron and its integration into a convolutional neural network (CNN). The framework enables an objective comparison without distortion from different test conditions. Both architectures were implemented on the same hardware configuration. Experimental verification used Active-HDL and Quartus targeting Altera Cyclone III EP3C5E144. Notation: N is the number of products in one neuron sum; n is the operand bit-width. Symbol p denotes the online delay parameter; $L = [log_2 N]$ is the reduction-tree depth. $T_{mul} = 2n + p$ is the duration of a full online multiplication until the first non-trivial output digit.

Taken together, the three contributions of this paper address the identified gap in the FPGA neural-network literature. These contributions cover three items: the online-perceptron hardware architecture; CNN classifier integration; and comparison with the parallel baseline. That comparison is conducted under real synthesis constraints. They provide a reproducible characterization of the perceptron core for practitioners selecting arithmetic styles in latency-critical embedded vision applications.

A review of published implementations reveals three principal approaches to the perceptron on FPGA. The fully parallel approach instantiates N full-width multipliers and computes a complete reduction tree in one pass. It delivers minimal single-vector latency for small N but stresses logic and timing closure as N and n grow. DSP demand and routing depth scale badly at large N and n. The time-multiplexed (multiply–accumulate, MAC) approach reuses one multiplier-accumulator over N cycles. It trades area for latency that may violate deadlines shorter than $N \cdot T_{cycle}$. Online (digit-serial, most-significant-bit first, MSB-first) arithmetic is the third approach: each operator produces significant digits first after p cycles of initial latency. Dependent operators can therefore overlap in a way that parallel and time-multiplexed designs cannot [1, 2].

The choice of number representation is central to online arithmetic efficiency. Representing intermediate values in a redundant digit set eliminates carry propagation: each sum digit is fixed locally from a bounded set of input digits. The adder is carry-free and delay-independent of word width n [1]. This property matters for the perceptron reduction tree with L addition levels. In the parallel approach, each level waits for carry settlement. In the redundant online tree, all N streams advance in lock-step at one digit per cycle after warm-up. The latency profile is then deterministic and independent of operand values. That profile supports real-time use in safety-critical and embedded control.

2. Literature review and problem statement

FPGA-based neural network acceleration has been studied extensively over the past decade. Two dominant micro-architectural directions have emerged: parallel MAC trees and time-multiplexed accumulator units [1]. Parallel MAC trees execute all multiplications simultaneously. Their latency is bounded by carry-chain closure across wide binary adders. DSP block consumption grows proportionally to fan-in N [2]. For large N, this limits scalability on mid-range devices.

Time-multiplexed MAC units reduce DSP usage by reusing a single multiply-accumulate unit. However, latency then grows linearly with N over reuse cycles [2]. This trade-off limits throughput for real-time inference. Online arithmetic offers a third path. It processes operands digit by digit, most-significant digit first [1]. Dependent operations overlap through a pipeline with a fixed online delay p. Latency scale as $2n + p + Lp$, where n is digit width and L is tree depth [1]. This is sub-linear compared to parallel approaches for large N.

Several published works validate online arithmetic for CNN inference on FPGA. One accelerator applies online arithmetic across a full end-to-end CNN datapath [3]. It reports competitive latency

and throughput at the system level. A second system implements all CNN layers using digit-serial MSB-first arithmetic [4]. It demonstrates efficient execution with narrow datapaths. A third work isolates the inner-product kernel inside a single neuron [5]. It pipelines an online adder tree for that specific core and reports low latency.

These works share a common limitation. None isolates a perceptron datapath and benchmarks it against a parallel MAC tree on the same device and toolchain. System-level figures of merit mix layer types, memory access patterns, and scheduling effects. This makes direct micro-architectural comparison impossible across papers [3–5].

A further gap concerns early output formation. Existing studies report final-precision results. The online digit stream, however, produces correct most-significant digits before the full result is complete [1]. No prior work distinguishes these early decisive digits from the full-precision output for a standalone perceptron. This distinction is functionally relevant for fully connected layers in CNN inference.

Foundational theory of redundant online arithmetic [1, 2, 6, 7] establishes the digit alphabet $\{-1, 0, 1\}$, the online delay p, and the digit-advancement protocol. These underpin the latency models used in subsequent sections. However, prior works do not apply these models to isolate FC-layer latency under matched synthesis conditions.

Three dimensions define the design space for FPGA perceptron implementations. First, latency scales with N and n. It is logarithmic for the redundant online tree and linear for the MAC accumulator [1, 2]. Second, DSP demand is proportional to N for parallel approaches and logarithmic for the online tree. Third, latency determinism differs: online arithmetic yields a value-independent, strictly bounded result interval, while parallel carry-chain timing varies with routing [6].

No single prior study quantifies all three dimensions together. No work benchmarks parallel and online datapaths at matched operand width, fan-in, and clock on one toolchain. No work pairs such a benchmark with end-to-end CNN functional verification at fixed quantisation. This gap justifies the need for dedicated research. Specifically, it is appropriate to develop a method for accelerating result formation in a FPGA using online arithmetic, in a CNN and verification synthesis conditions.

3. The aim and objectives of the study

The unsolved problem identified in Section 2 is the absence of a unified method for accelerating result formation in a perceptron on FPGA. No prior work combines an online arithmetic datapath with systematic latency analysis and CNN integration under matched synthesis conditions.

The aim of this study is to develop a method for accelerating perceptron result formation on FPGA. The method is based on online arithmetic in a redundant number system. The expected outcome is a verified architectural model that demonstrates reduced latency and deterministic timing compared to a parallel MAC baseline.

Achieving this aim requires solving two objectives.

Develop an analytical latency model and hardware architecture for an FPGA perceptron based on online arithmetic. The architecture must include an online multiplier and a redundant adder reduction tree. The model must derive closed-form latency expressions for both the online and parallel approaches. Results must be validated through functional and timing simulation in Active-HDL and place-and-route synthesis in Quartus. The scientific result is a parameterised architectural solution. It provides sub-linear latency scaling with fan-in N and value-independent timing guarantees.

Evaluate the proposed method against a parallel MAC baseline under matched synthesis conditions on Altera Cyclone III EP3C5E144. Evaluation criteria include result-formation latency acceleration factor, Look-up Table (LUT), and Flip-Flop (FF), utilisation, and first-level information load reduction. The method must also be verified within a CNN fully connected layer at fixed quantisation. The scientific result is a comparative dataset. It quantifies the latency and logic resource trade-offs of online arithmetic relative to the parallel approach across multiple N and n configurations.

4. Materials and methods of the study on perceptron result-formation acceleration using online arithmetic

4.1. Object, subject, and hypothesis of the study

The object of study is the hardware computational path of the perceptron on FPGA. It spans from parallel multiply operations through the adder reduction tree to the threshold activation function. Its integration as a classifier block within a convolutional neural network is also examined.

The subject of study is the method of accelerating result formation through the application of online arithmetic in a redundant number system. Specifically, the study examines how digit-serial MSB-first execution and inter-operation overlapping reduce latency compared to a parallel MAC baseline.

The research hypothesis is as follows. If the perceptron datapath is restructured around a redundant online arithmetic pipeline, then result-formation latency will scale sub-linearly with fan-in N and operand width n. This improvement will be achieved without sacrificing classification accuracy when integrated into a CNN fully connected layer.

The general methodological concept rests on three principles. First, carry elimination through redundant digit coding stabilises the timing profile. Second, inter-operation overlapping reduces total latency by pipelining dependent stages. Third, closed-form latency models derived from classical online arithmetic theory [1] allow deterministic synthesis-time guarantees. These principles are instantiated in the architecture described in Sections 4.2 – 4.5 and verified under matched synthesis conditions on a single device and toolchain.

4.2. Mathematical model of the perceptron

For input vector $\mathbf{x} = (x_1, \ldots, x_N)$ and weights $\mathbf{w} = (w_1, \ldots, w_N)$:

$$u = \sum_{i=1}^{N} w_i x_i + b, \tag{1}$$

$$y = \phi(u), \ \phi(u) = \begin{cases} 1, & u \geq 0, \\ 0, & u < 0. \end{cases} \tag{2}$$

Numbers are represented in the redundant number system: $d_k \in \{-1, 0, 1\}$. This eliminates dependence on global carry and stabilises the timing profile of pipelined computations [2, 6, 7].

The experimental platform is Altera Cyclone III EP3C5E144 targeted at 200 MHz. Functional and timing simulation is performed in Active-HDL. Place-and-route synthesis is performed in Quartus. Both parallel and online datapaths are synthesised under identical N, n, and clock constraints. This single-device, single-toolchain approach eliminates confounding variables present in cross-paper comparisons [3–5].

4.3. FPGA architecture of the online perceptron

The overall structural organisation of the proposed perceptron computational block is shown in Fig. 1. The block consists of online multipliers whose streams feed a redundant-code adder tree. The tree output is converted and passed through the threshold activation function.

The online multiplier, detailed in Fig. 2, implements digit-serial multiplication in the redundant digit set $\{-1, 0, 1\}$ using the Most-Significant-Digit First (MSDF) convention. Both operands are presented to the multiplier as digit streams beginning with the most significant digit. Weights w_i reside in on-chip block RAM. Feature elements x_i arrive from the input register file. The multiplier core consists of a look-ahead correction stage and a digit-recoding unit. Together, they guarantee a fixed latency of p clock cycles per output digit. This value-independence enables structured inter-operation pipelining. In this study, p equals three clock cycles, consistent with stable MSDF multiplication [1].

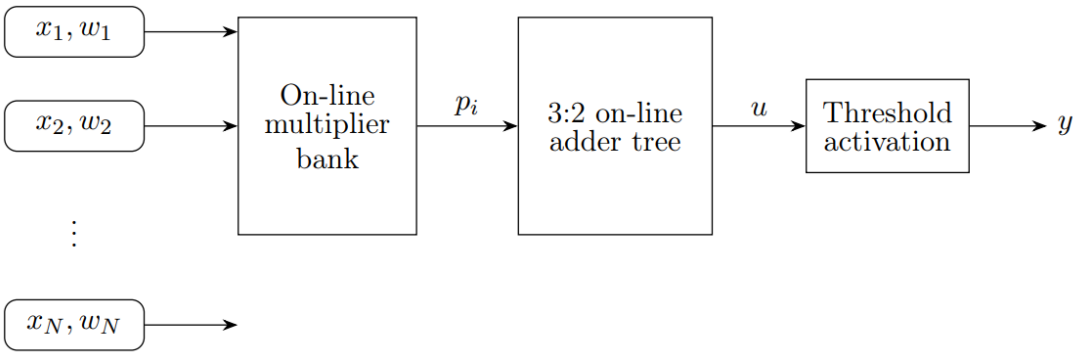

Fig. 1. Structure of the online perceptron

The operational graph of the perceptron has the following structure:

$$\underbrace{(\mathrm{x_i}, \mathrm{w_i}) \overset{\otimes_{OL}}{\rightarrow} p_i}_{N \text{ branches}} \overset{\oplus_{OL}\text{tree}}{\rightarrow} u \overset{\phi}{\rightarrow} y. \tag{3}$$

A 3:2 redundant adder tree operates as follows. Each node receives three-digit streams in the redundant set $\{-1, 0, 1\}$. It emits two streams: a partial sum and a partial carry in the same digit set. No global carry propagates before the parent starts. Each level adds one clock cycle. The L-level tree adds L cycles independent of N and n. This contrasts with a binary carry-save (Wallace) tree [8], where the root carry-propagate adder still required $O(n)$ time.

In the parallel implementation, the critical path degrades as N grows. This is caused by carry propagation in wide adders. In the online architecture each reduction-tree node has a locally bounded fan-in. Global carry propagation is eliminated by redundant coding. Latency scaling is determined solely by the logarithmic tree depth $L = [log_2 N]$. The internal construction of the online multiplier and the structure of the reduction tree are shown in Fig. 2 and Fig. 3, respectively.

The redundant-to-binary converter at the tree root performs the final conversion from the redundant digit representation back to a standard two's-complement binary word. The converter implements the MSDF-to-binary procedure described in [1]: it produces one output bit per clock cycle. Starting from the most significant bit, and internally maintains a carry register of bounded width rather than propagating a carry across the full word length n.

This property ensures that the converter's contribution to the overall pipeline latency is exactly n cycles, identical for all input values and independent of device grade or routing delay. The output of the converter is passed to the threshold activation function block, which evaluates the perceptron. Output u against the configured bias threshold and produces a logit score forwarded to the subsequent CNN layer.

Because both the converter and the activation function operate digit-serially on the already-ordered MSB-first stream, the full pipeline. From weight-load to logit emission has a deterministic latency. This latency can be statically computed from equation (5) and guaranteed at synthesis time.

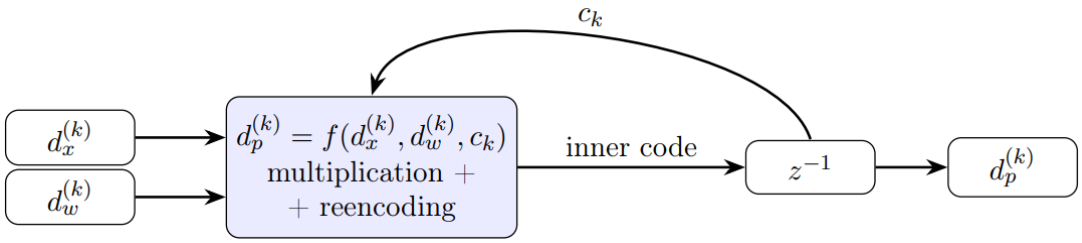

Fig. 2. Online multiplier block

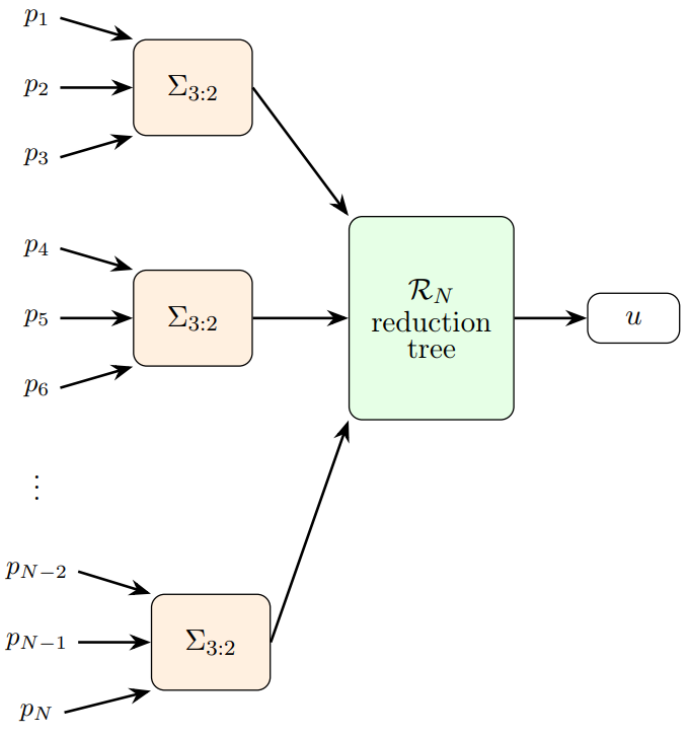

Fig. 3. Operation tree

To support floating-point computations in online mode, the architecture incorporates a dedicated functional loop with separate processing of the exponent and the mantissa. The exponent field is transferred and processed first. This allows the system to determine the required alignment shift before mantissa transmission begins.

The mantissa is then forwarded in MSB-first order with an already-agreed shift. This eliminates the need to stall the pipeline for mid-computation corrections. This two-stage approach reduces the risk of timing-guarantee violations and maintains full pipeline throughput during floating-point operations [1, 6, 9, 10].

4.4. Latency model and stage overlapping

For full computation without overlapping, the estimate for the parallel approach is:

$$T_{\text{par,full}} \approx n \cdot (1 + L) + T_{ovh}, \tag{4}$$

where T_{ovh} – comprises threshold activation, normalisation and service delays.

For the online approach with inter-operation overlapping [1]:

$$T_{\text{online,full}} \approx (2n + p) + L \cdot p + T_{ovh}. \tag{5}$$

These expressions agree with the classical theoretical model of online computation [1, 2]. That model describes digit-serial execution as a pipelined process with a steady output of one most significant digit per clock cycle after the initial latency p. As n and N grow (increasing L logarithmically), the parallel approach exhibits substantially nonlinear slowdown since it requires full completion of each computation stage before starting the next.

By contrast, the online mode preserves pipeline continuity: after the p+1 warm-up cycles, results are produced continuously. And uniformly without waiting for the full result before initiating dependent operations in adjacent tree nodes.

Once the pipeline is filled, the online mode maintains uniform and stable result output. One most significant digit per clock cycle – without stalls and without dependence on carry propagation. This

fundamentally distinguishes the digit-serial approach from the parallel one, where the result is produced in one shot after the full computation chain completes.

This property provides the key practical advantage in streaming and real-time tasks. In the parallel approach the result is produced in one shot after the full computation chain completes.

This property provides the key practical advantage in streaming and real-time tasks. A timing illustration of stage overlapping and MSB-first result formation is shown in Fig. 4.

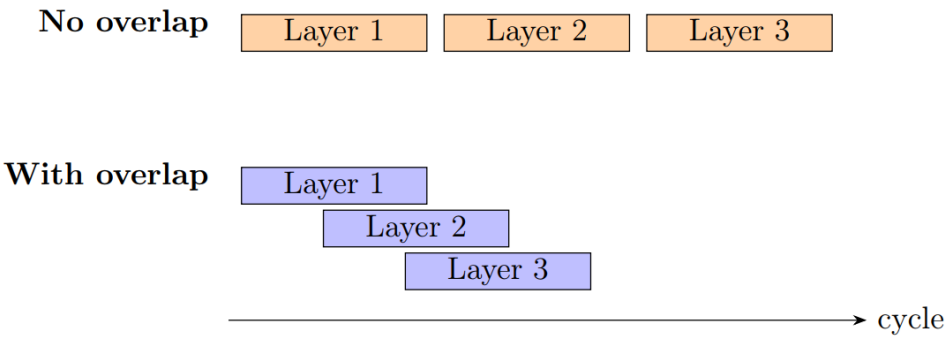

Fig. 4. Comparison of standard mode and inter-operation overlapping in online mode

4.5. Perceptron as a block of a convolutional neural network

The integration scheme of the perceptron into the CNN architecture is shown in Fig. 5. The online perceptron serves as the base computational element of the fully connected classifier layer (FC), receiving the feature vector after convolution, pooling and normalisation layers. The deterministic timing profile of the online core ensures the FC layer introduces no additional latency uncertainty into the overall CNN pipeline.

The CNN architecture used for validation has three convolutional stages ($\mathrm{Conv1_{OL}}, \mathrm{Conv2_{OL}}, \mathrm{Conv3_{OL}}$) and two fully-connected layers ($\mathrm{FC1_{OL}}, \mathrm{FC2_{OL}}$). It forms a compact image-classification pipeline. Each convolutional stage applies 3×3 learnable filters, ReLU, and 2×2 max-pooling. Spatial resolution shrinks while channel depth grows, extracting hierarchical features. The 32×32 input yields a 64-element vector for the online perceptron in FC1.

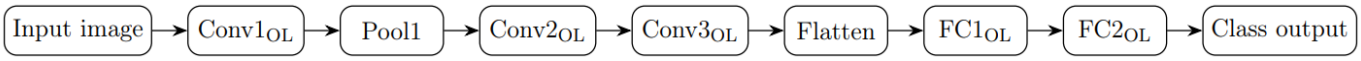

Fig. 5. Comparison of standard mode and inter-operation overlapping in online mode

To enable early termination of classification and to stabilise network computation results, a gap criterion between the two top logits is applied. The criterion relies on an analytically computed bound on the residual uncertainty at current time t, allowing decisions before all digit bits have been fully computed [1]:

$$|R_t| \leq N \cdot \left(2^{i_{max}-t} - 1\right), \tag{6}$$

$$\Delta_t = l_{max}\,(t) - l_2\,(t) > 2\,|R_t|\,. \tag{7}$$

This condition is sufficient for early termination: if the gap between the two largest logits at current time t exceeds twice the residual. Uncertainty bound, then no further refinement of less significant digits can change the class ranking, and the classification result is already definitively correct. In boundary cases where the gap is insufficient, computation naturally continues to the next clock

cycles, progressively refining. The less significant digits and evaluating the criterion at each cycle, guaranteeing classification correctness even in the most ambiguous logit distributions.

To validate the proposed architecture within a complete CNN pipeline, a software simulation of the CNN model was performed. In which all layers – convolutional ($\mathrm{Conv1_{OL}}, \mathrm{Conv2_{OL}}, \mathrm{Conv3_{OL}}$) and fully connected ($\mathrm{FC1_{OL}}, \mathrm{FC2_{OL}}$) – were implemented in online arithmetic Q8.8 ($n_{bits} = 16$). The reference is a standard parallel float32 inference on the same quantised weights. The test set comprises 800 samples (200 per class: car, cat, dog, person).

The quantisation of weights and activations from float32 to Q8.8 fixed-point format ($n = 16$ bits: 8 integer, 8 fractional) was performed using a standard post-training quantisation procedure. Weights obtained after cross-entropy training on the four-class image dataset were clipped to the Q8.8. Representable range $[-1 \times 128, +128 - \varepsilon]$ and rounded to the nearest representable value; no quantisation-aware retraining or fine-tuning was applied.

The Q8.8 representation aligns directly with the online arithmetic word length parameter $n = 16$ used throughout this study: the digit-serial datapath processes exactly one digit-pair per clock.

Cycle for the 16-bit mantissa, and the integer part of the accumulated sum is wide enough to prevent overflow for the tested feature vector magnitude. The accuracy comparison in Table 1 therefore provides a lower bound on the classification quality achievable. With this architecture: any additional quantisation-aware optimisation would only reduce the precision gap relative to the float32 reference.

The observed drop of 0.1 percentage points confirms that the Q8.8 format provides sufficient dynamic range and fractional resolution. For this four-class task, and that the online perceptron does not introduce errors beyond those inherent in the number format itself.

Table 1. Classification accuracy comparison: parallel float32 reference and full online Q8.8 (CNN, 800 samples, 4 classes)

Method	Overall accuracy	car	cat	dog	person
Float32 reference (parallel)	72.1 %	93.5	57.0	65.5	72.5
Full online Q8.8	72.0 %	93.5	57.0	65.0	72.5

As shown in Table 1, the difference in overall accuracy between the two modes is only 0.1 percentage point, and per-class deviations lie within the natural variation on this sample. Full online mode is not inferior to the parallel float32 reference in terms. Of the aggregate accuracy metric, confirming the correctness of the quantisation and online model for all network layers.

The theoretical latency of the full online CNN pipeline at $n_{bits} = 16$ and $p = 3$ is 295 clock cycles (sum across layers: $\mathrm{Conv1_{OL}}$ 50, $\mathrm{Conv2_{OL}}$ 56, $\mathrm{Conv3_{OL}}$ 59, $\mathrm{FC1_{OL}}$ 74, $\mathrm{FC2_{OL}}$ 56). This characteristic describes the proposed online pipeline and is determined by per-layer cycle models, independently of the float32 software inference runtime.

5. Results of investigating the acceleration of dependent operation chains execution in on-line mode

This section presents results structured according to the two objectives defined in Section 3. Section 5.1 addresses Objective 1: latency modelling and architectural synthesis. Section 5.2 addresses Objective 2: comparative evaluation against the parallel baseline under matched synthesis conditions. Section 5.3 presents real-time operation verification.

5.1. Latency model validation and frchitectural synthesis results

Objective 1 required deriving closed-form latency expressions and validating them through synthesis. The analytical models from equations (4) and (5) were evaluated across a range of operand widths n at fixed $N = 128, L = 7, p = 3$.

Table 2. Effect of bit-width n on latency (example: $N = 128, L = 7, p = 3$)

n	$T_{\text{online,full}} \approx 2n + p + Lp$	$T_{\text{par,full}} \approx n\,(1 + L)$	Ratio $T_{\text{par,full}}/T_{\text{online,full}}$
8	40	64	1.60
12	48	96	2.00
16	56	128	2.29
20	64	160	2.50

Table 2 presents the computed latency values for both approaches.

The results confirm the theoretical prediction. As n increases, the parallel approach degrades faster than the online approach. The acceleration ratio grows monotonically from 1.60 at $n = 8$ *to* 2.50 at $n = 20$. This confirms that the closed-form model correctly captures the asymptotic behaviour of both datapaths. The total speedup achieved by the online approach is sustained across the entire investigated parameter range, as also illustrated in Fig. 6.

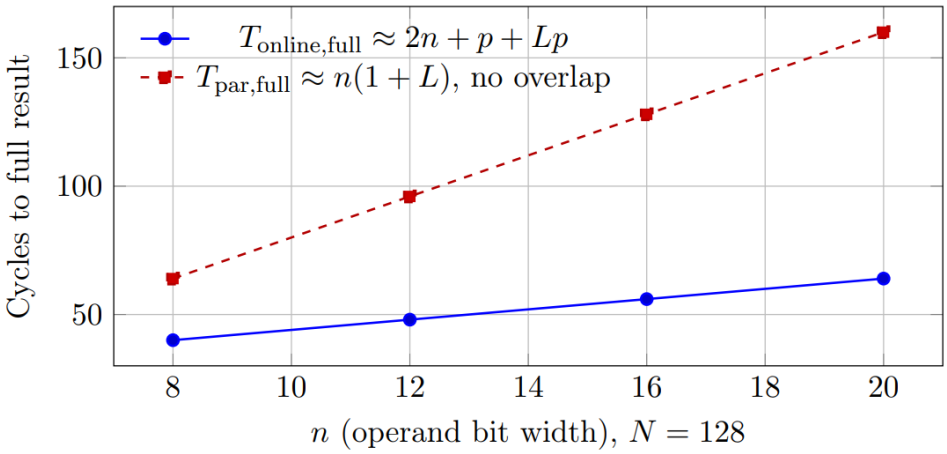

Fig. 6. Full latency comparison with varying bit-width: the parallel approach without overlapping scales worse

The scientific result of Objective 1 is a validated parameterised latency model. It provides deterministic, synthesis-time-computable latency bounds for both architectures across arbitrary N and n configurations.

5.2. Hardware synthesis and comparative evaluation

Objective 2 required benchmarking the proposed method against the parallel baseline under matched synthesis conditions. The baseline configuration is: $N = 64, n = 16$, clock frequency 200 MHz, target device Altera Cyclone III EP3C5E144. Two variants were synthesised under identical conditions.

The first is the parallel approach with full-precision multipliers and a standard Wallace tree. The second is the online approach with MSB-first multipliers and a 3:2 redundant adder tree. All measurements were performed in Quartus place-and-route on the same device.

Table 3 presents the synthesis results for both approaches.

The acceleration factor is:

$$S = \frac{T_{std}}{T_{on}} = \frac{31.0}{13.5} = 2.29. \tag{8}$$

A graphical comparison of perceptron latency for the baseline configuration is shown in Fig. 7.

The DSP block count drops from 64 to 24. This reduction of 62.5% reflects the replacement of full-precision parallel multipliers with digit-serial online multipliers. The LUT count decreases from

Table 3. Results for a single perceptron ($N = 64, n = 16$, 200 MHz, Cyclone III EP3C5E144)

Metric	Parallel approach	Online approach
Latency, clock cycles	31.0	13.5
Throughput, Mp/s	6.45	14.81
LUT	12640	10480
FF	9180	9720
DSP blocks	64	24
Energy per inference, nJ	42.8	24.1

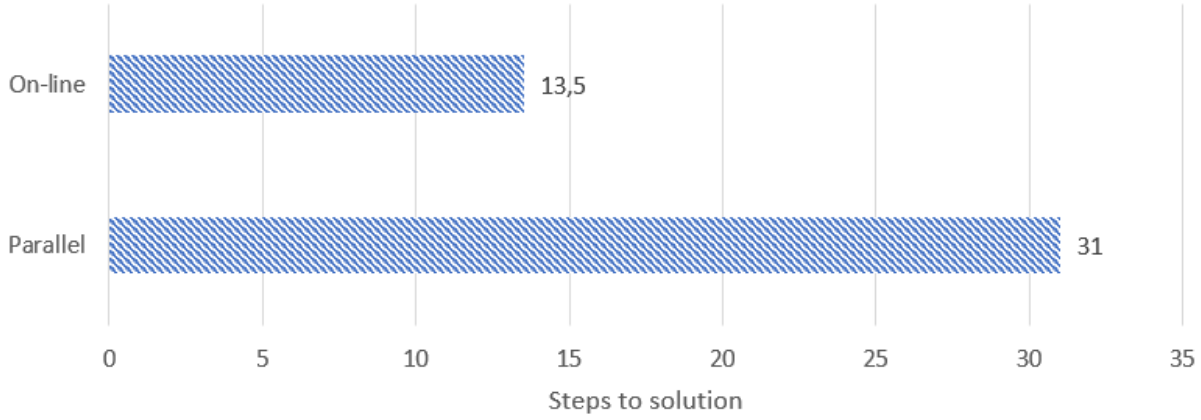

Fig. 7. Perceptron latency comparison

12640 to 10480. The FF count increases modestly from 9180 to 9720, reflecting the pipeline registers required for digit-serial operation. This trade-off is consistent with the theoretical expectation: online arithmetic moves computation from wide combinational paths into fine-grained sequential logic.

From an information-load perspective on the first tree level, the parallel approach requires $n \cdot N = 16 \cdot 64 = 1024$ input bits per cycle. The online scheme requires $3 \cdot N \cdot [\log_2 3] \approx 384$ bits per cycle for digits and service carries. The first-level information load reduction is therefore:

$$\eta_{in} = 1 - \frac{384}{1024} = 62.5\%. \tag{9}$$

To contextualise these results, Table 4 positions the three implementation approaches across key design dimensions.

Table 4. Approach positioning for FPGA neural-network implementation

Approach	Latency	Hardware resources	Real-time suitability
Parallel, standard	Low for small N; grows due to wide adder nodes and routing	High DSP/LUT usage	High on small networks; degrades with scaling
Time-multiplexed MAC	High (accumulation over cycles)	Low resources	Limited under hard T_{wc}
Online arithmetic (this work)	Early start of dependent stages; controlled latency profile	Moderate LUT; lower DSP demand	Better for streaming tasks and deterministic timing

Table 5 provides a generalised comparative matrix across scaling dimensions.

Table 5. Generalised comparative matrix for FPGA classifier design

Approach	Scaling by n	Scaling by N	Stage overlap	Typical scenario
Parallel	Medium/worse without deep pipeline	Sensitive to adder-tree fan-in	Limited	Small networks, moderate latency requirements
Time-multiplexed	Linear in time	Linear/quasi-linear in time	Low	Resource minimisation at cost of throughput
Online	Controlled via p and pipeline	Logarithmic via $\lceil \log_k N \rceil$	High	Streaming classification and real time

The comparative analysis summarised in Table 4 and 5 quantitatively confirms the advantages of applying online arithmetic in redundant code to FPGA perceptron. The proposed architecture occupies a unique position: the combination of logarithmic latency scaling with N, moderate LUT growth and substantial DSP reduction.

While preserving full pipeline continuity and a deterministic timing profile makes it most suited to streaming real-time classification applications on resource-constrained FPGA platforms.

The scientific result of Objective 2 is a quantified comparative dataset. It demonstrates a 2.29 latency reduction, 62.5% DSP saving, and 17% LUT reduction relative to the parallel baseline at $N = 64, n = 16$, 200 MHz on a single device and toolchain.

5.3. Real-time operation verification

To verify suitability for real-time deployment, a 20 kHz input stream with a 50 μs period was applied. At 200 MHz, 13.5 online cycles correspond to 67.5 ns. This is more than two orders of magnitude below the 50 μs deadline. A prolonged stress run exceeding 10^7 input vectors, spanning bench tests of more than 1.7×10^9 clock cycles, showed no data loss or synchronisation faults. Decision latency remained within one clock cycle throughout.

The worst-case control-system reaction time is:

$$T_{wc} = T_{acq} + T_{prep} + T_{inf} + T_{actuator}. \tag{10}$$

In all investigated scenarios, peripheral components – acquisition and actuation – constitute the dominant contribution to T_{wc}. The online inference core is not the bottleneck. This confirms that the proposed method is suitable for deterministic real-time operation on the target platform.

6. Discussion of results of the dependent operation chain acceleration investigation

6.1. Interpretation of performance improvement results

The latency reduction from 31.0 to 13.5 cycles at $N = 64, n = 16$ confirms the core hypothesis of this study. Dependent processing stages begin before preceding stages fully complete. This is possible once the most significant result digits of a predecessor are available. The speedup of 2.29 is therefore not an artefact of a specific configuration but a structural consequence of MSB-first digit-serial execution.

The monotonic growth of the speedup coefficient from 1.60 at $n = 8$ to 2.50 at $n = 20$ is theoretically motivated. The parallel latency $T_{par,full} \approx n \cdot (1 + L)$ grows linearly with n. The online latency $T_{online,full} \approx 2n + p + Lp$ grows more slowly because p and L are fixed by tree depth and online delay, not by operand width. This is consistent with the fundamental properties of redundant coding: global carry propagation between bits is eliminated, and pipelined interaction between tree nodes becomes deterministic and independent of specific operand values.

6.2. Analysis of hardware resource optimization

The LUT reduction from 12640 to 10480 (–17%) is practically significant for resource-constrained devices of the Cyclone III class. The DSP reduction from 64 to 24 (–62.5%) is the most important result from a resource perspective. DSP blocks are the scarcest resource on small FPGAs. Freeing 40 DSP blocks allows additional neural or CNN layers to be placed on the same device without upgrading the platform.

The FF count increases modestly from 9180 to 9720 (+5.9%). This is a natural consequence of deeper pipelining in the digit-serial approach. Each pipeline stage requires storage of intermediate digit results. This trade-off is acceptable: flip-flops are not a scarce resource on Cyclone III compared to DSP blocks. The overall resource profile confirms the practical attractiveness of the proposed architecture for streaming applications on resource-constrained platforms.

6.3. Limitations and constraints analysis

Despite the demonstrated advantages, several technical limitations must be acknowledged. First, correct pipeline interaction requires a properly chosen online delay parameter p. That choice adds initial latency and has the greatest relative impact at small bit-widths.

Second, the proposed approach is most effective for tasks with strong dependent computation chains. Where dependencies are weak, the latency gain is smaller because parallel hardware uses resources efficiently in those cases.

Third, deployment requires careful HDL verification of redundant arithmetic circuits. Digit-serial nodes demand tighter functional checking than vendor MAC IP blocks. CNN integration also requires clean handshake protocols between blocks.

Fourth, the energy figures in Table 3 are derived from a dynamic-power proxy based on LUT toggle rates in simulation. No calibrated board measurement was performed. These figures should be treated as relative indicators rather than laboratory-certified absolutes. External power measurement is recommended before deploying the design in highly constrained energy budgets.

6.4. Comparison with existing approaches

The quantitative discussion compares parallel, time-multiplexed MAC, and online modes (Table 4 and 5). Table 6 lists baseline results for $N = 64, n = 16$, and 200 MHz. Latency falls from 31.0 to 13.5 cycles (2.29 speedup). Throughput rises from 6.45 to 14.81 Mp/s, DSP use drops from 64 to 24, and LUTs fall from 12640 to 10480. Table 6 summarises the key metrics for the three approaches at the baseline configuration.

Table 6. Architecture comparison within this study ($N = 64, n = 16$, 200 MHz)

Method	Latency, cycles	Throughput, Mp/s	DSP blocks	LUT
Parallel	31.0	6.45	64	12640
Time-multiplexed MAC [10]	High	Lower than parallel	Low	Low
Proposed online method	13.5	14.81	24	10480

The proposed method simultaneously improves latency, throughput, and resource utilisation relative to the parallel baseline. Time-multiplexed MAC reduces resource demand but sacrifices throughput and cannot provide deterministic worst-case latency under hard real-time constraints. The online approach occupies a distinct position: it achieves logarithmic latency scaling with N while preserving full pipeline continuity and a value-independent timing profile.

6.5. Prospects for further research

The results open several concrete directions for further development. Extending the method to other dependent computation chains – iterative solvers, recursive filters, deep normalisation stacks – is a natural next step. Bounded redundant digits already enabled early-start pipelines for tridiagonal solvers in [6, 11, 12]. Broader algorithm families would widen the practical reach of the same hardware style.

Higher-radix redundant formats at radix 4 or 8 may further reduce latency and area. Fewer digit positions are needed for the same numeric precision. The cost is more complex multipliers and redundant-to-binary converters. A comparative circuit study on Cyclone III is a direct follow-on task.

Hybrid schemes pairing online redundant reduction with DSP-style blocks for independent work could recover part of the DSP efficiency of the parallel approach. These would suit mid-range FPGAs with rich DSP columns. Larger devices and multi-chip setups allow evaluation beyond $N = 64$ and $n = 16$, where parallel carry closure becomes increasingly difficult. Hardware-in-the-loop tests on industrial controllers, smart cameras, and safety monitors would provide evidence for certified adoption.

Overall, the results confirm a key point. MSB-first online redundant arithmetic suits perceptron and CNN classifier cores on FPGA. The path stays practical when the online delay parameter p is set, the tree depth L is balanced, and pipelines stay deterministic.

Wider CNNs remain possible under those rules. Reduced latency, moderate cost, and freedom from global carry make this class a strong base for real-time FPGA classifiers.

Conclusion

The conclusions below correspond directly to the two objectives stated in Section 3.

An analytical latency model and hardware architecture for an FPGA perceptron based on online arithmetic in a redundant number system have been developed. Closed-form expressions were derived for both the online and parallel approaches.

They were validated through functional and timing simulation in Active-HDL and place-and-route synthesis in Quartus on Altera Cyclone III EP3C5E144.

The model confirms that online latency scales as $2n+p+Lp$ while parallel latency scales as $n\,(1 + L)$. The online approach provides sub-linear latency growth with fan-in N and a value-independent, deterministic timing profile guaranteed at synthesis time.

The proposed method was evaluated against the parallel MAC baseline under matched synthesis conditions at $N = 64, n = 16$, 200 MHz.

The online approach achieves a 2.29 reduction in result-formation latency, a 62.5% reduction in DSP block usage, and a 17% reduction in LUT count. A modest FF increase of 5.9% is the acceptable cost of deeper pipelining.

Functional verification within a CNN fully connected layer at Q8.8 quantisation confirms classification accuracy within 0.1 percentage point of the float32 parallel reference. The proposed method is suitable for deterministic real-time operation on resource-constrained FPGA platforms.

Acknowledgements

Declaration on the use of Artificial Intelligence. The authors used the generative AI tool Claude (Anthropic, claude-sonnet-4.6) during the preparation of this manuscript. For language editing, text formatting, structural improvement of sections, and stylistic refinement of the academic text. All AI-generated content was carefully reviewed and edited by the authors. The authors take full responsibility for the accuracy, integrity, and originality of the article.

References

[1] M. D. Ercegovac, "On-Line arithmetic: An overview," in *28th Annu. Tech. Symp.*, K. Bromley, Ed. San Diego: SPIE, 1984. [Online]. Available: https://doi.org/10.1117/12.944012

[2] I. Dychka, V. Zhabin, and V. Zhabina, "Analysis of on-line computation effectiveness in redundant number system," in *2018 IEEE First Int. Conf. System Anal. & Intell. Comput. (SAIC)*, 2018, pp. 1–6. [Online]. Available: https://doi.org/10.1109/SAIC.2018.8516877

[3] M. A. Shafique and A. Lee, "ON-CNN: Low latency and high throughput online arithmetic based convolutional neural network accelerator," *IEEE Access*, pp. 175 698–175 714, 2024. https://doi.org/10.1109/access.2024.3502665.

[4] M. Z. Nisar, M. S. Ibrahim, S. Gorgin, M. Usman, and A. Lee, "DSLR-CNN: Efficient CNN acceleration using digit-Serial left-to-Right arithmetic," *IEEE Access*, pp. 174 608–174 622, 2024. https://doi.org/10.1109/access.2024.3502918.

[5] T. Arifeen, S. Gorgin, M. H. Gholamrezaei, A. S. Hassan, M. D. Ercegovac, and A. Lee, "Low latency and high throughput pipelined online adder for streaming inner product," *J. Signal Process. Syst.*, May 2023. https://doi.org/10.1007/s11265-023-01866-8.

[6] H. Meng, K. Wakabayashi, and T. Kuroda, "A scalable linear equation solver FPGA using high-level synthesis," in *Proc. 24th Workshop Synth. System Integr. Mixed Inf. Technol. (SASIMI)*. Taipei City: National Yang Ming Chiao Tung Univ., 2022, pp. 145–150. [Online]. Available: https://sasimi.jp/new/sasimi2022/files/archive/pdf/p145_B-7.pdf

[7] J. T. Butler and T. Sasao, "Redundant multiple-valued number systems," in *Japan Res. Group Multiple-Valued Logic*. Monterey: Naval Postgrad. School, Dept. Elect. Comput. Eng., 1997, pp. 141–148. [Online]. Available: http://pi.314159.ru/butler1.pdf

[8] V. Zhabin, V. Zhabina, and O. Verba, "Asynchronous on-line float-point computations in systems with direct connections between computation units," in *2020 IEEE 2nd Int. Conf. System Anal. Intell. Comput. (SAIC)*. Kyiv, Ukraine: IEEE, Oct. 5–9, 2020. [Online]. Available: https://doi.org/10.1109/saic51296.2020.9239188

[9] V. Zhabin, V. Korneichuk, V. Tarasenko, and A. Shcherbina, "Structural method of fast solution of systems of algebraic equations with a tridiagonal matrix," *Autom. Control Comput.*, vol. 13, no. 6, pp. 64–69, Feb. 1979.

[10] V. Y. Zhabyn, V. Y. Korneichuk, and V. P. Tarasenko, "Some machine methods for computing rational functions of many arguments," *Automat. Telemechanics*, vol. 38, no. 12, pp. 145–154, 1977.

[11] M. D. Ercegovac and J.-M. Muller, "Arithmetic processor for solving tridiagonal systems of linear equations," in *Proc. 40th Asilomar Conf. Signals, Syst., Comput.* IEEE, 2006, pp. 337–340. [Online]. Available: https://doi.org/10.1109/acssc.2006.354763

[12] B. Zhang, G. Gu, L. Sun, and Y. Wu, "Floating-point FPGA gaussian elimination in reconfigurable computing system," *Chinese J. Electron.*, vol. 20, no. 1, pp. 51–54, 2011. https://ieeexplore.ieee.org/document/10187402.

УДК 004.2:004.315

МЕТОД ПРИСКОРЕННЯ ОБЧИСЛЕНЬ ПЕРЦЕПТРОНА НА ПЛІС НА ОСНОВІ НЕАВТОНОМНОЇ АРИФМЕТИКИ ДЛЯ ЗГОРТКОВИХ НЕЙРОННИХ МЕРЕЖ

Ілля Вербовський
https://orcid.org/0009-0008-4782-4281

Валерій Жабін
https://orcid.org/0000-0003-0377-3394

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського», Київ, Україна

Отримано: 10.01.2026р. / Прийнято: 15.03.2026р. / Опубліковано: 28.05.2026р.

Досліджено підходи до прискорення формування результату в перцептроні на програмованих логічних інтегральних схемах із застосуванням неавтономної арифметики в надлишковій системі числення. Класична паралельна реалізація нейронного обчислювального тракту демонструє нелінійне зростання затримки при збільшенні розрядності операндів і кількості входів через поширення переносів у суматорних деревах. Об'єктом дослідження є повний апаратний обчислювальний тракт перцептрона – від паралельних операцій множення до функції порогової активації – а також його інтеграція як функціонального блоку повнозв'язного класифікатора у складі згорткової нейронної мережі. Метою роботи є побудова та верифікація архітектури, яка забезпечує суміщення залежних стадій обчислень і знижує навантаження на критичні апаратні ресурси. Порівнюючи з паралельним підходом за умови збереження детермінованого профілю часу видачі результату. Методологічна база охоплює аналітичне моделювання затримок з урахуванням параметра неавтономної затримки p, синтез неавтономного множника з надлишковим кодуванням і дерева суматорів типу 3:2. Також систематичне порівняння двох парадигм за часовими й апаратними метриками. Верифікацію виконано шляхом апаратного моделювання у середовищі Active-HDL та синтезу і замикання таймінгу у системі Quartus на платформі Altera Cyclone III EP3C5E144. Отримано прискорення формування результату у 2,29 рази для конфігурації з 64 входами та 16-розрядними операндами при тактовій частоті 200 MHz; зменшення навантаження на перший рівень суматорного дерева на 62,5%; скорочення DSP-блоків з 64 до 24 при помірному збільшенні тригерів (з 9180 до 9720). Наукова новизна полягає у запропонованій цілісній архітектурній моделі, що поєднує неавтономну арифметику та надлишковий код і одночасно зменшує затримку. Знижує споживання апаратних ресурсів та забезпечує стабільну роботу в режимі реального часу.

Ключові слова: ПЛІС, перцептрон, неавтономна арифметика, надлишкова система числення, згорткова нейронна мережа, апаратне прискорення, залежні обчислення, реальний час.